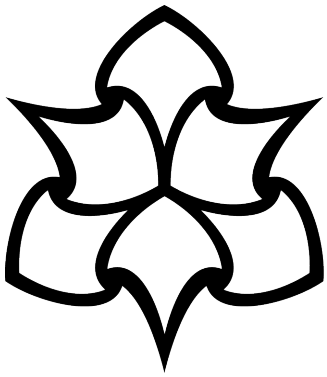


Journal Pre-proof

Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN

Raja Majid Ali Ujjan, Zeeshan Pervez, Keshav Dahal, Ali Kashif Bashir, Rao Mumtaz, J. González

PII: S0167-739X(19)31833-3
DOI: https://doi.org/10.1016/j.future.2019.10.015
Reference: FUTURE 5240

To appear in: *Future Generation Computer Systems*

Received date : 12 July 2019
Revised date : 18 September 2019
Accepted date : 27 October 2019

Please cite this article as: R.M.A. Ujjan, Z. Pervez, K. Dahal et al., Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN, *Future Generation Computer Systems* (2019), doi: https://doi.org/10.1016/j.future.2019.10.015.

- This paper presented a sFlow and adaptive pooling based sampling approach to efficiently detect distributed denial of service attack for IoT applications.
- The programmability of SDN enables flexible placement of intrusion detection capabilities such as Snort and data sampling.
- In the data-plane of SDN to reduce the processing and network overhead at switches, we deployed sFlow and adaptive polling based sampling individually.
- In control-plane, to optimize detection accuracy, we deployed Snort IDS collaboratively with Stacked Autoencoders (SAE) deep learning model.
- The empirical evaluation of the proposed work demonstrated higher detection accuracy for sFlow as compared to adaptive pooling based approach.

Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN

Raja Majid Ali Ujjan[a], Zeeshan Pervez[a], Keshav Dahal[a], Ali Kashif Bashir[b], Rao Mumtaz[c] and J. González[c]

[a]School of Computing, Engineering and Physical Sciences, University of the West of Scotland, Paisley, PA1 2BE, UK
[b]Department of Computing, Mathematics, and Digital Technology, Manchester Metropolitan University, Manchester M1 5GE, UK
[c]GS LDA, Aveiro, Portugal

ARTICLE INFO

Keywords:
DDoS
IoT
SDN
Snort
Sampling

ABSTRACT

Distributed Denial of Service (DDoS) is one of the most rampant attacks in the modern Internet of Things (IoT) network infrastructures. Security plays a very vital role for an ever-growing heterogeneous network of IoT nodes, which are directly connected to each other. Due to the preliminary stage of Software Defined Networking (SDN), in the IoT network, sampling based measurement approaches currently results in low-accuracy, higher memory consumption, higher-overhead in processing and network, and low attack-detection. To deal with these aforementioned issues, this paper proposes sFlow and adaptive polling based sampling with Snort Intrusion Detection System (IDS)and deep learning based model, which helps to lower down the various types of prevalent DDoS attacks inside the IoT network. The flexible decoupling property of SDN enables us to program network devices for required parameters without utilizing third-party propriety based hardware or software. Firstly, in data-plane, to lower down processing and network overhead of switches, we deployed sFlow and adaptive polling based sampling individually. Secondly, in control-plane, to optimize detection accuracy, we deployed Snort IDS collaboratively with Stacked Autoencoders (SAE) deep learning model. Furthermore, after applying performance metrics on collected traffic streams, we quantitatively investigate trade off among attack detection accuracy and resources overhead. The evaluation of the proposed system demonstrates higher detection accuracy with 95% of True Positive rate with less than 4% of False Positive rate within sFlow based implementation compared to adaptive polling.

1. Introduction

The rapid growth of the Internet of Things (IoT) has become very popular throughout the world, it has emerged in our modern smart homes, vehicles and many other wearable gadgets. IoT is a combination of large interconnected devices, such as household appliances, wearable devices, medical devices, smart vehicles and public facilities provider [41, 31]. According to the research [64], tens of billions of vulnerable devices will be connected to IoT. Most of these devices do not use secure communication protocols or appropriate security measures when interfacing with computing and storage units. These vulnerabilities provide temptation not only for enterprises wanting to acquire data for the adoption of intelligent management system or digital proof but also for malicious users (i.e., attackers), which diffuse intrusion or service disruption such as Distributed Denial of Service (DDoS). Once a DDoS attack occurs then it can severely threaten human life safety directly or indirectly - for instance, connected medical devices, home or critical infrastructure monitoring. Current studies in [58, 30] depict that IoT is also vulnerable to viruses. Studies of current DDoS attacks have also proved that IoT has ubiquitous loopholes with the initial stage. If the security approaches have not been deployed in IoT then DDoS attacks unwillingly arise. DDoS attacks are capable to make most of the network services unavailable by utilizing all available server resources with continuous undesirable traffic floods. Cisco Visual Networking Index (VNI) has depicted in a survey, nearly 17 millions of DDoS intrusion activities can happen by 2020, as a triple increment to 2015 incidents [11]. Nature of attacks is also switching from a single flooding attack to multiple attack vectors.

In 2016, the largest DDoS attacks [42]were recorded, these DDoS attacks were caused due to unauthorised remote access to IoT devices and security weakness. This vulnerability enables the undetected attackers to instal Botnet at various nodes of IoT devices, where compromised IoT devices nodes generate the high traffic and exhausts the resources.

Most of financial, social-media, entertainment, medical, business and engineering sectors are under attacks [22], thus posing reputation and financial losses of these sectors at risk. However, it has become a very mandatory concern to detect this kind of attacks in real time with larger network premises. In large networks, Intrusion Detection Systems (IDS) are widely deployed for providing a safeguard against network threats. IDS is capable to monitor and analyze network packets either in-line mode or passive mode. Most of the detection systems utilize collaborative work with IDS and algorithms for accurate detection processing and reliable data analysis. IDS cannot be a feasible solution for real-time monitoring without these capabilities, which left systems vulnerable for various attackers [4]. The major concerns behind this fact is that network traffic is exponentially growing day by day, so it is becoming more complex to select ap-

raja.vjjan@uws.ac.uk (R.M.A. Ujjan); zeeshan.pervez@uws.ac.uk (Z. Pervez); keshav.dahal@uws.ac.uk (K. Dahal); dr.alikashif.b@ieee.org (A.K. Bashir); shmu@gs-lda.com (R. Mumtaz); jonathan@gs-lda.com (J. González)
ORCID(s):

propriate IDS placement point, addition to this IDS is also limited with hardware resources within large networks. The survey of recent study [55] depicts performance comparison between Snort and machine learning applications, experimental results show that Snort as a stand-alone tool reaches 9.5% average packet drop with 4 Gbps network and 20% average packet drop within 10 Gbps network. To optimize network packet losses several studies have been proposed such as sampling-based IDS in SDN [18] and deep learning based DDoS detection system in SDN [45].

Software Defined Networking (SDN) in collaboration with packet sampling and deep learning based applications have received much attention from researchers [17, 45, 18]. SDN provides the centralized global view of the entire infrastructure with programmable control-plane and data-plane individually. As intrusion detection technology is exponentially rising in current network security approaches, researchers are facing challenges on how to manage a large significant amount of sampled traffic load in an effective way. The researchers have proposed network traffic characteristics which comprise the size of flow table [5], deviation in traffic flow [26] and anomaly statistics with sampling [33]. The authors in [5] proposed sampling technique based on flow-size. According to the authors, most of the network attacks use the small size of traffic flow from a malicious traffic source. From experiments, they depicted that malicious traffic flow size is much smaller than certain sampled threshold values with constant probability. The authors of [26] proposed an IDS based method on statistics of traffic flow. The research studies represent that flow rapidly increased during the event of a network attack. The sampled traffic was partitioned into sub-sections followed by a source autonomous system data. Moreover, their experimental analysis pragmatically improves the overall intrusion detection ability. In [33], the authors presented that sampling approaches reduce detection ability for non-volume dependent anomalies. The authors implemented three different intrusion detection algorithms for detecting non-volume malicious traffic from both sampled traffic and original data trace. Their results depicted that sampling network traffic can increase false-positive triggers and reduce overall detection capability of IDS. The authors of [20] proposed a feasible pattern matching approach to improve Snort IDS performance, the research work mainly focused to reduce false-positive alarms. The deployment of IDS was based on co-detection of misuse and anomaly approaches. The overall system proposed fast and reliable packet inspection with very fewer resources utilization. Most of the authors have proposed DDoS detection approaches in SDN with unreal and redundant datasets, which mainly focus only on SDN control-layer. The existing literature comprises control-plane sampling approaches due to the initial stage of SDN features, which also lead to low detection accuracy and higher resource consumption.

To solve the aforementioned problems, we proposed a novel solution to effectively detect DDoS traffic in IoT. Our proposed solution uses sFlow and adaptive polling based sampling in data-plane to manage heavy traffic flows. These sampling approaches lower down network burden and enables IDS to record malicious activities effectively. Our work uses Snort and Stacked Autoencoders (SAE), an unsupervised algorithm to improve detection accuracy with real-time network traces collected with IDS in SDN. In this work, we utilized packet based sFlow and time-oriented adaptive polling based sampling approaches inside data-plane switches, to rebuild real-time network flows statistic with low CPU and network overhead. This enables the DDoS detection model to effectively classify traffic during DDoS floods. We also provided a comparison between OpenFlow based sFlow and adaptive polling based sampling schemes to improve the IDS detection inaccuracies, while keeping total sampled flows lower than IDS processing power. This enables to acquire data with Snort IDS over the controller. Our work achieved less than 4% of False Positive alerts with 95% True Positive rate in packet-based sampling, which is comparatively better than time oriented sampling. Proposed work is evaluated in Tensorflow 1.4 by utilizing a confusion matrix.

This paper is organised as: in Section II, we present related work. Section III provides main sampling approaches. In the Section IV, we discuss main architectural components of the proposed system. In Section V, we present traffic analysis. Finally, we discuss experimental setup and results evaluation in Section VI and VII respectively.

2. Related work

The researchers have performed a significant amount of work with intrusion detection technologies on traditional network infrastructure[65]. Most of the research is carried out with detection of DDoS and mitigation of DDoS attacks within legacy network [3]. Comparing to traffic demands of future networks, legacy networks are found significantly expensive and least secure with a different perspective of traffic analysis and deployment. To address these challenges, SDN infrastructure with OpenFlow protocol has successfully improved various security challenges [56]. Despite providing modern solutions in SDN, control-plane and data-plane is not extensively explored with real-time datasets to detect new types of DDoS attacks. Due to centralized network activity management and programmability approaches, SDN networks could become vulnerable for potential attacks [28]. In particular, attacking SDN control-plane with DDoS floods could run down a larger portion of network [62]. In data-plane, detecting DDoS attacks is difficult due to the fact that OpenFlow enabled switches have the least information due to isolated traffic flows, this weakness makes the network more vulnerable to attacks. Attackers can launch DDoS traffic with simple knowledge, and available tools and hardware assets [24].

In existing research, only preliminary research is carried out for optimising the security of IoT with SDN technology. The authors in [15] designed a distributed secure architecture based in SDN for the IoT domain. Similarly, authors of [2] also investigated preliminary discussion for how to mitigate the DDoS attacks OF IoT devices, the authors have utilised the SDN based sampling approaches for identifying

the anomalies, the major aim was to collect the switches information for enhancing security accuracy. Moreover, the authors of [44] also designed a hybrid security countermeasure system to provide a safeguard against link spoofing attacks inside SDN based IoT controller.

Many researchers have proposed well known DDoS detection approaches in SDN context. The authors in [6] utilized Self Organizing Maps (SOM), a machine learning (ML) model to detect DDoS attacks by utilizing the features of traffic flow. In this method, authors presented six tuples of feature extraction such as Average of Packet per flow (APf), Average of Bytes per flow (ABf), Average of Duration per flow (ADf), Percentage of Pair flows (PPf), Growth of Single flow (GSf), and Growth of Different Ports (GDP). Flow is divided into sub-components if there exists a larger connection in time context then all six features of this work are modified to be used in identical flows. The major aim was to detect DDoS flooding attacks inside kernel based modular detection engine. Based on traffic features, detection engine classified in benign traffic and malicious traffic entries. The proposed work can only perform with table flow and cannot be extended to detect DDoS attacks in heavy traffic flow streams inside switches. Similarly, authors in [39] also utilized ML approaches to analyse and differentiate network traffic flows. In [7], authors used four malicious traffic detection algorithms in SDN contexts, such as Threshold Random Walk with Credit-Based (TRW-CB), Rate-Limiting, Maximum Entropy Detector and Network Traffic Anomaly Detector (NETAD). According to authors TRW-CB algorithm and Rate-Limiting algorithm were capable to detect every first packet arriving with new sessions, then collected packets were sent to the controller for deep inspection to identify attack event. The Maximum Entropy Detector was able to collect every initial packet then consistently collect these packets by every predefined t seconds, this helps to make traffic class distribution to classify normal traffic distribution class with the help of maximum entropy estimation. The authors in [56] depicted malicious traffic mitigation AVANT-GUARD approach. The mitigation performed only at control-plane traffic. Authors modified data-plane with the help of OpenFlow for utilizing two new models such as connection mitigation model and actuating trigger model.

In [51] SDN provides the various DDoS defence mechanisms via programmability feature, in the proposed solution of SD-IoT framework, the majority of IoT controllers of the pool are responsible for the centralised logic, which handles the overall IoT networks. The centralised logical controller is easy to maintain and manage but it also carries some security vulnerabilities. Like SDN based networks, SD-IoT is also responsible for creating detection and mitigation approaches by utilising programmability features.

Generally, most of ML approaches differentiate attacks traffic flows followed by certain feature characteristics, which are obtained from that traffic. Many researchers have utilized ML-based anomaly detection approaches within small-sized networks, but in large networks, acquiring network flows and feature statistics manipulation provides significant overhead in SDN controller [17], according to the authors, controller response time plays a vital role in attack detection scenarios. In general, ML-based anomaly detection techniques rely on trained datasets in the model. Moreover, authors of [17] proposed anomaly detection and mitigation techniques by employing sFlow based data gathering in the SDN platform. The proposed model consists of a collector unit, anomaly detection unit and anomaly mitigation unit. The collector unit acquires all incoming traffic flows with sFlow technique. After flow acquisition, an entropy-based algorithm differentiates traffic flows with benign class and malicious class. After malicious traffic identification, anomaly detector utilizes IP addresses and port numbers, which helps to mitigate the attacks. In contrast to anomaly detection, some researchers have proposed a statistical solution to detect DDoS traffic [61, 54]. The authors of [38] provided an entropy-based solution for early stage DDoS detection mainly at POX SDN controller. This proposed solution carries some limitations such as, if hosts number increases then model triggers a higher false-positive rate. To lower down the network overhead such computation process from various hosts, the authors of [63] proposed fast and effective entropy-based detection method for abnormal traffic, which works with traffic flows. The proposed approach utilizes scheduling based single queue process with k numbers of queues with logical subsets, these were individually assigned to network switches. In the event of heavy traffic request at controller, the system utilized logical queues to manage traffic requests with the sequential scheduling process.

According to [40] IoT, dynamic environment requires access control from a centralised controller, that is responsible for managing traffic flows and to restrict the unauthorised access from IoT network nodes. Controller access control helps to simplify the nodes authentication on each IoT endpoints nodes for reducing network and computation overhead.

In [7], the authors utilize maximum entropy estimation methodology to detect the benign traffic class to resolve network security issues inside home and office-based SDN networks. The authors performed the experiment with OpenFlow switches in POX controller, where the proposed technique only worked with low rate network traffic traces, mainly focused to detect attacks inside networks. Addition to this, other authors also used entropy detection to catch port-scans and worm-propagation attacks [17]. Moreover, anomaly-based DDoS detection approach was also proposed by [61]. This research work significantly lowers down control-plane overhead carried out in the SDN based edge switches.

Most of the legacy network routing protocols are limited for designing an IoT infrastructure due to lack of its computational and energy efficient approach. According to the authors of [29], DDoS attacks are the most common security threats, especially in IoT based infrastructures. Most of IoT gateways for traffic collection can become a single point of failure for other IoT sensors. These types of vulnerabilities generate DDoS attacks in IoT. The authors of [50] proposed another distinctive solution which is based on packet_in fil-

tering methodology. This approach utilizes the contents lists extracted from the header field of the packet_in event of the SDN controller. However, if the attacker launches new attack flows which do not match to predefined content lists, then the proposed approach failed to detect abnormal activities. To improve the aforementioned issue, the authors in [13] depicted an attack detection approach. This proposed solution locate all infected interfaces, which are used by intruders to infiltrate and ex-filtrate network sources. Generally, anomaly detection approaches use fixed threshold values. If some deviations occur in incoming abnormal statistical features, then it can easily identify malicious traffic. The authors of [37], proposed detection solution to count TCP SYN flooding traces in SDN. The main aim of authors is to effectively mitigate flooding attacks with SDN programmable countermeasures. The proposed solution deployed with OpenDaylight extension module, where it monitors all TCP connections and block the infected hosts.

Research has also carried out to accomplish network security in the SDN with traffic sampling. Researchers have mainly focused to process a large amount of traffic effectively, they depicted various solutions with traffic characteristics, including flow information, flow size and entropy changes in flow [17]. The authors of [32] proved that sampling with detection traffic results in massive false positive detection rate within non-volume dependant anomalous traffic. The authors utilized three various detection algorithms to catch non-volume-based traffic flows separately with sampled data and original data of the model. The results showed a low detection rate with false alerts, overall model detection performance was degraded. To overcome this issue, the authors of [21] suggested a new improved solution with new pattern matching technique in co-detection of Snort IDS sensor. The authors merged misuse detection and anomaly detection techniques to lower down false alarms. The proposed method caters an effective and fast packet inspection by utilizing low IDS resources comparatively to legacy systems. Recently, the authors in [10] depicted a high precision DDoS detection solution with SDN based Xboost classifier. In this method, most of DDoS attack mechanism is analyzed to provide an effective solution. ICMP floods, TCP SYN floods and UDP floods were sent to POX controller via grab bag tool, all these attacks were manipulated using grab bag connection records for evaluation in DDoS classifier. Table 1 precisely depicts major contributions of authors in SDN to detect DDoS traffic.

From the above literature, we identify most of the research has been carried out for the DDoS detection only at SDN control layers with redundant unreal datasets, rather than at network data layers. There has been a significant work deployed to classify heavy traffic DDoS attacks into real data traces.

The Internet of Things (IoT) is always under consideration for adopting security and privacy requirements to create personal profiles. With the adoption of Personal Network and access to the specific function of advanced APIs, it is also capable to utilises third parties devices on Layer 2 for security consideration. the IoT network can easily become vulnerable due to multiple points of IoT nodes. Addition to this, the lack of computational and memory resources constraints, which are associated with DDoS attack and anomaly detection mechanism rely on traffic sampling approaches to detect real-time DDoS attack detection accuracy. The major aim of our research is to focus the DDoS detection with the processing of large packets of IoT network.

In contrast, we proposed co-detection model for IoT, firstly, we utilize distributed traffic sampling in network data-plane to maintain heavy network flows, secondly, the model utilizes Snort IDS parallel to deep learning model to adopt higher detection accuracy with low false triggers. In data-plane, the appropriate sampling rate is employed inside all network switches so that IDS can differentiate malicious and benign traffic during heavy flow. This model caters analytical evaluation for detection accuracy, which is mainly based on sFlow and adaptive polling based sampling. The sampling approaches in collaboration with IDS can adopt effective detection, which is a significant improvement over existing work. Primarily, these methodologies require less processing due to rectified data features. Due to low network overhead, the proposed model is feasibly capable to be used in various scales to meet security requirement in modern networks.

3. Background Overview

This paper mainly focuses on IoT DDoS attack detection model by first utilising sFlow and adaptive polling sampling with Snort IDS to identify the network traffic, and then classify the traffic with Stacked AutoEncoder (SAE) into benign and malicious traffic. In the following, we present the building block of our proposed DDoS attack detection.

3.1. Stacked AutoEncoder (SAE) in SDN

The SAE comprises several auto-encoders, its structure consists of three layers, input or visible layer, hidden layer, and reconstruction or output layer. The data inputs are fed into a visible layer. The reconstruction layer generates output. SAE algorithm is unique in nature as compared to CNN, DBN, and RBM deep neural networks. Firstly, SAE is made up of simple and straightforward structure, it is trained in very less time as compared to other mentioned Deep Neural Network (DNN) algorithms [45]. Secondly, SAE does not use labelled datasets due to the nature of the unsupervised learning approach. In contrast, CNN is based on supervised learning, whereas, DBN and RBM utilize semi-supervised datasets. Finally, the SAE algorithm also utilizes the outputs as inputs, and detailed features from flows can be extracted with good training approach in SAE. This paper utilizes detailed features of a dataset based on the SAE approach to improve the detection rate of DDoS attacks in SDN. SAE as deep neural networks (DNN) utilizes sparse auto-encoders and soft-max classifier for extraction and classification of unsupervised datasets. A sparse AutoEncoder is a neural network that uses three layers, where input and output layers work with P nodes and hidden nodes with Q nodes. The M nodes from input layers show records with P features such

Table 1
Existing literature survey for DDoS detection in SDN.

Author	Year	Description	Methods
[6]	2010	Proposed work utilizes six SDN based traffic features to detect DDoS attacks.	Neural network model,SOM.
[7]	2010	Proposed method utilizes collected flow entries with fixed time intervals at controller.	Periodic flow based detection
[9]	2010	Packet-in Bloom filter used in switch memory to manage the DDoS traffic.	Load balancing, Bloom filter
[60]	2011	this work utilizes SDN TE application for effectively detection in networks.	SDN TE, load balancing.
[34]	2011	Depicted various traffic collection and detection approaches to detect DDoS.	TRW-CB, maximum entropy, Rate limiting.
[59]	2012	Block-based neural network (BBNN) used with IDS to detect DDoS in FPGA.	BBNN algorithm, IDS,
[56]	2013	Proposed method lower down the burden between control-plane and data-plane.	Interface mitigation.
[21]	2013	Utilized Snort with combined techniques of anomaly detection and misuse detection.	Anomaly, misuse detection, Snort IDS.
[17]	2014	Flow based feature extraction to detect worm propagation, DDoS, portscan attacks.	Entropy detection.
[18]	2015	Heavy traffic request handled using scheduling based traffic scenarios.	multiple direction for malicious data
[61]	2015	Detection model employed in edge switch to reduce traffic flows at controller.	Entropy based detection
[14]	2016	Utilized semi-supervised SVM to detect anamolies in SDN.	SVM, anamoly detection
[38]	2016	Quantitative Shannon entropy utilized to classify early stage DDoS attacks.	Shannon method of entropy
[13]	2016	A heavy DDoS attack rate is detected via Sequential Probability Ratio Test .	SPRT method.
[46]	2017	Deep learning based, multi-vector DDoS traffic monitoring and detection system.	Deep learning, SAE algorithm
[37]	2017	Counter measure mitigation implemented as SLICOTS at controller for TCP SYN floods.	SLICOTS methods
[10]	2018	Extreme gradient algorithm proposed to classify DDoS attacks with low false triggers.	Machine learning.

as $X = \{x_1, x_2, x_3, \ldots, x_p\}$. The training function manipulates the output layer as the input layer. The main objective of sparse auto-encoders as network is to calculate feasible values for the weighted matrices like $\cup \in \sigma e^{q\times p} = (WW^T)$ and $\cup^{'} \in \sigma e^{p\times q} = (W^{'T}W^{'})$. The bias vectors are given as $b_v =\in \sigma e^{Q\times 1}$, similarly, $b_v^{'} =\in \sigma e^{P\times 1}$. We set a learning identity approximation function $X^{\wedge} \approx X$ by using back propagation algorithm [43]. There is a large number of activation functions used for hidden and output nodes but we utilized Sigmoid activation function to activate the function of $s\cup, b_v$, which is formulated in the following equations:

$$J(W) = s\cup, b_v = s(\cup X + b_v) = \frac{1}{1 + e^{-}(\cup X + b_v)} \quad (1)$$

$$J(W) = \frac{0.5}{r} \sum ||P(X)_n - P(X^{\wedge})_n|| + 0.5\lambda(WW_T + W^{'T}W^{'}) \quad (2)$$

$$J_{sparse}(W) = J(W) + \beta \sum_{J=1}^{q} d_{KL}(\rho||\rho_J^{\wedge}) \quad (3)$$

In equation 2, we optimize SAE generalization ability to get feasible features representation of datasets. For this objective we added cost function to achieve feasible weight learning in sparse SAE, backpropagation is utilized for minimization. In this equation term of $\frac{0.5}{r} \sum ||P(X)_n - P(X^{\wedge})_n||$ represents the all data inputs average of sum-of-square errors records with corresponding model output values with number of r records. The second term of $0.5\lambda(WW_T + W^{'T}W^{'})$ is used for weight decay with the help of λ to maintain over fitting inside our model. In equation 3, β represents the weight coefficient of sparsity penalty with Kullback-Leibler divergence function. This divergence function is used to manipulate low average activation values, this function reaches the minimum values of 0 when $\rho = \rho_J^{\wedge}$, among them ρ is sparsity coefficient and $\rho_J^{\wedge}$ is known as average activation of

Jth hidden node during training input. The divergence function of Kullback-Leibler (KL) is provided in equation 4 as below:

$$KL((\rho||\rho_J^{\wedge}) = \rho \log \frac{\rho}{\rho_J^{\wedge}} + (1-\rho) \log \frac{(1-\rho)}{(1-\rho_J^{\wedge})} \quad (4)$$

3.2. The sFlow Sampling Approach

To deal IDS aforementioned issues which are depicted in Section II, we implemented packet sampling with sFlow capability in our proposed design. The sFlow sampling technique creates mandatory flow statistics, which is normally used for manipulation purpose [17]. Due to this nature sFlow approach gathers sampled packets and then rebuilds the new flow patterns by updating packet counters for each flow entries in OpenFlow (OF) enabled controller. The sFlow collector propagates all related traffic statistics to malicious detection model. The sFlow approach enables data-plane forwarding logic in more efficient and aggregate way, which reduces network overhead from all deployed infrastructure. This approach of sFlow does not rely on specific flow entries but the major aim is to reduce the number of flow entries up to the relevant level for detection model.

According to [48], sFlow is a suitable technology to monitor real-time and virtual networks. sFlow agents are utilized in network switches to collect switch statistics with the required sample rate then transfer to common sFlow collector. Normally, mechanism of sampled data gathering comprises either replicating real packetâĂŹs header or feature extraction from network packets, then sample packets after being decoded transferred to cache flow record, every flow record is updated and saved to look up each flow record entry. Following by protocol information such as FIN flags, timeout, inactivity or once cache memory is full then records are wiped out from its cache memory and transferred to traffic analysis application. However, in sFlow case, all monitoring, decoded hash flow, flush functionality are wiped out from switches then these are propagated to centralized sFlow analyses also known as a collector, this runs on an individual server with abundant resources to support a large number of request flows with network stabilization.

Moreover, the sFlow helps to overcome possible table size flow limitation which is mandatory for specific OF hardware deployed infrastructure [17]. Meanwhile, the sFlow data collection technique gathers appropriate statistics for the effective anomaly detection process. As sFlow collector gathers packet samples consistently, it also updates relevant counters for monitoring module with a specific time window. Therefore, each flow entry does not require further maintenance for manipulating detailed flow statistics, this enables to lower down the complexities for flow collection models. In Section VI, low CPU and resources usage of the model will be presented to support sFlow mechanism in our proposed work.

3.3. Adaptive Polling Sampling Approach

In proposed work, adaptive polling sampling algorithm is used to refine polling intervals based on the rate of change of each flow. If switches flow rapidly changes then polling interval is reduced, on contrast, if polling intervals increased then flow remains stable. The polling interval becomes unchanged when flow fluctuates. We dynamically manipulated polling frequency by predicting future flow rates on the basis of historical data. The polling interval is calculated with the help of predicted future flow rates rather than the current live flow rate. We utilized the combined approach of proportional linear prediction (PLP) and weighted linear prediction (WLP) to estimate the next flow rate. This approach in the SDN environment is also discussed in [8]. Firstly, we implemented a low pass filter [19] to estimate the future flow rate. Low pass filter for flow rate prediction is provided in equation 5.

$$Z_{(X+1)} = \beta C + (1-\beta)W \quad (5)$$

In equation 5, $Z_{(X+1)}$ represents the predicted values for flow rate *pf* next upcoming polling, *C* is currently calculated value and W depicts two different current polling values such as $0 \leq \beta \leq 1$, based on the network load, it is implemented in experiment. Afterwards, next flow rate is calculated with PLP given as below:

$$Z^P_{(X+1)} = Z_X(1 + \frac{\delta(Z_X - Z_X)}{Z_{(X-1)}}) \quad (6)$$

$$Z_{(X+1)} = \beta Z^P_{(X+1)} + (1-\beta)Z_X \quad (7)$$

Overall, aggregated rate of change in flow is provided as below in equation: 8:

$$M = \frac{|Z_{(X+1)} - Z_X|}{Z_X} \quad (8)$$

In equation 8, M represents overall flow changes where, two threshold values of M_{min} and M_{max} used with $0 \leq M \leq 1$ conditions. If the value of $M < M_{min}$ then detected flow gradually changes as compared to current conditions. When $M \geq M_{min}$ then it brings increment in polling interval thresholds, and detected flow rate rapidly changes in contrast to current predictions, it leaves polling threshold values slow down here. Other than these depicted conditions, the detected flow rate becomes proportional to current predictions and leaves polling thresholds unchanged.

The adaptive polling algorithm to sample SDN switches is depicted in algorithm 1. This algorithm uses UB_{max} and LB_{min} as upper bound and lower bound intervals respectively. ΔT_N is polling intervals for N_{th} intervals numbers, in the algorithm, the line 3 and 4 represent overall flow which is currently active in switches with no sampling rate adjustment. Afterwards, we utilized the equations 2, 3, 4 to get the next predicted sample rate with the rate of change of flow in line 7. The given algorithm line 8 to 11 depicts adopted polling intervals rules. The line 8 and 9, if $M < LB_{min}$, represent that polling interval is higher, similarly if $M \geq UB_{max}$, pragmatically depicts the polling interval is reduced shown in line 10, 11. Finally, line 11 and 12 leave the overall polling intervals unchanged.

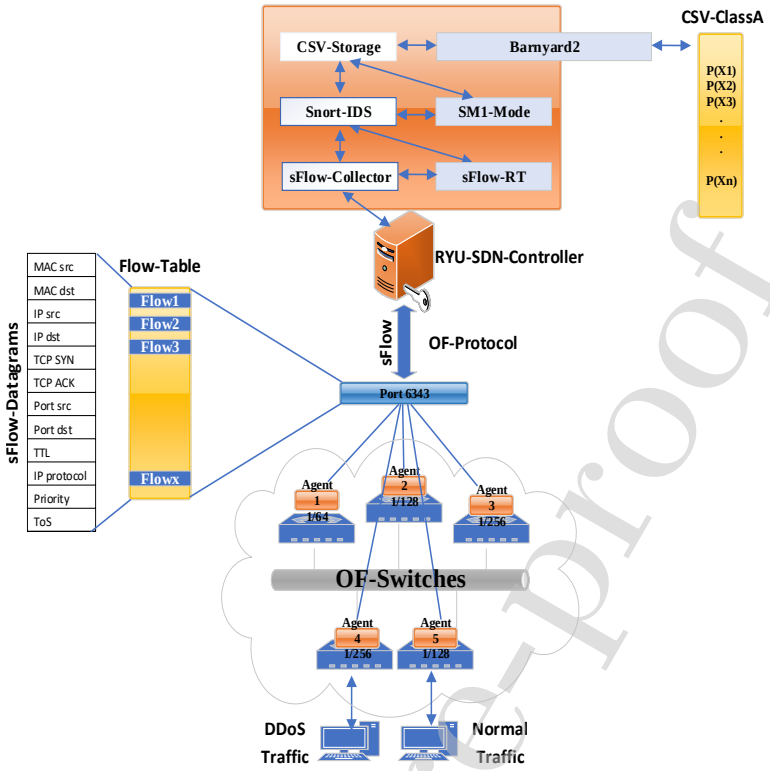


Figure 1: Data Acquisition with sFlow Sampling.

Algorithm 1 Adaptive polling sampling in SDN switches

Input: $M_{min}, M_{max}, UB_{max}, LB_{min}$, polling intervals, active flows, historic flows
Ensure Output: $Time_{int}$, next polling intervals
if *active flow in packet − in − event*, **return** False
function DO NOT SEND (*flow to Switches table*)
  else if *polling table* $\Delta T_{sec_t} = Curr_{sec_t} - Pre_{sec_t}$ **return** True
  Utilize equation 6 , 3, *to calculate next flow rate*;
  Utilize equation 8 *to compute aggregate flow change*;
  if $M < LB_{min}$ then do;
  $T_{Intervals} = min(T_{intervals} + 1, T_{max})$;
  else if $M \geq UB_{max}$ then do;
  $T_{Intervals} = max([T_{intervals}/2], T_{min})$;
  else $T_{intervals} = min(T_{intervals}, T_{max})$;
  end if

4. Proposed Design

In our proposed design, we mainly utilized open source tools and technology such as sFlow-RT [12] collector and Ntopng [47] collector for sFlow and adaptive polling based sampling respectively in data-plane. Snort as Network Intrusion Detection System (NIDS) with Barnyard2 is also used for creating a database to store datasets inside Ryu SDN controller.

4.1. sFlow Collector (sFlow Sampling)

In the sFlow sampling approach, we utilized three main modules consisting on *sFlow-collector*, *Snort-IDS*, and *Data-storage*. This section of the proposed design utilizes sFlow sampling and Snort IDS to manipulate flow classification for higher rate DDoS detection model. This approach collaboratively works with OpenFlow SDN controller and northbound interface communication, which is feasible to create new insertion policies for traffic engineering in Open vSwitch (OVS) switches. The sFlow module is used to apply packet flow sampling with individual sampling agents on each OVS switches. All sFlow agents deployed remotely via northbound APIs with sampling decision rates of 1/28, 1/64/, 1/256, 1/512, 1/1024 [49]. These sFlow agents standalone deliver collected sampled traffic streams with suitable sampling rate from all deployed switches towards sFlow-RT. The sFlow-RT is a real-time network monitoring tool that collects and analyze sampled streams in detailed visualized form. The sFlow-RT is an industry standard and helps to incorporates InMon asynchronous analytics[16]. The sFlow-RT [49] real-time analytics engine is capable to receive the continuous flow streams from agents of network-embedded switches. The sFlow-RT decodes the received flow from sFlow agents into actionable metrics via RestFlow API [52]. The RestFlow API is capable to customize the configuration of setting up thresholds, metrics manipulation and flow measurements. These API works with HTTP and rest calls with external applications, in addition to this embedded JavaScript and standardized ECMA scripts enable to work internally.

Table 2
Six Major DDoS attack types in our test-bed

DDoS Attacks	Sever based	Network based	Application based	Description
HTTP-DDoS floods			✓	Volumetric attacks with HTTP GET or HTTP POST request.
DNS-DDoS floods			✓	Attacks to infect DNS resolution, such as application or API.
ICMP-DDoS floods		✓		These attacks are ping floods, to overwhelm the ICMP-echo requests
UDP-DDoS floods		✓		These attacks disrupt the device processing ability.
Smurf-DDoS floods		✓		These attacks launch spoofed IP addresses to target.
IP-DDoS floods	✓			Attacks on server source and destination IP addresses

Overall, sFlow-RT provides detailed visible pipeline which helps to bring the sustainable performance of all hosts, servers, applications and embedded devices in a single global view. This helps to classify and analyze feasible network features so that the deployed model can work effectively.

4.2. Snort-IDS (sFlow Sampling)

Our paper uses Snort as NIDS to collect malicious traffic and benign traffic, We used various frameworks and tools eg. The Metasploit framework [35], hping3 [23]and Low Orbit Ion Cannon (LOIC) [1] to launch various DDoS attacks with customized TCP/IP packets, we utilize the hping3 tool to emulate customized DDoS attacks. Our model also uses the LOIC DDoS tool to launch TCP, UDP and ICMP floods. By utilising Metasploit and Kali Linux framework, LOIC, and hping3, we generated six most common types of DDoS attack, which are provided in Table 2.

The main purpose of the Metasploit framework is to generate malicious traffic with some payloads. Snort collects malicious traffic with SM1-mode and benign traffic is acquired using SM2-mode shown below in Table 3. Both benign and malicious traffic is manually emulated from hosts connected directly to OVS switches, attacking hosts are assumed to perform DDoS attack flooding with port scanning using hping3 and LOIC DDoS tool. We implemented a standalone rule set in Snort detection-engine configuration file for both modes mentioned with details in Table 3. In test-bed, once traffic is collected with deployed sFlow sampling agents, then all sampled traffic stream is mirrored with port 6343 to the sFlow-collector module. In Figure 1, sFlow-collector module is integrated with sFlow-RT. Due to the flexibility of SDN Ryu controller modular design, we utilized sFlow-collector module traffic into Snort-IDS module. In this module Snort is configured with packet logging mode, if traffic is emulated from malicious hosts then it utilizes SM1-mode and if traffic is coming from benign hosts then SM2-mode is used to record logs followed by Table 3.

4.3. Flow-collector (Adaptive Polling Sampling)

In this Flow-collector module, we deployed a time-driven sampling mechanism with adaptive polling intervals shown in Figure 2. In this section, the polling scheme enables to fetch timely accurate switches statistics on a fine-grained level. Our polling mechanism instructs all active switches stream flows by giving limited bandwidth channels and time intervals. It is mandatory to adjust all active flows polling intervals. The main objective of this polling based sampling is to focus, if the number of flows is exponential rising then

Table 3
Snort Traffic acquiring modes with sFlow sampling.

Description	Snort Two Modes	
	SM1 (Malicious)	SM2 (Benign)
TCP rules	15 signatures	8 signatures
UDP rules	10 signatures	6 signatures
ICMP rules	6 signatures	6signatures
CSV Database	Barnyard2	Barnyard2
Log file	snort.log.xx1	snort.log.xx2
Traffic Monitor	sFlow-RT	sFlow-RT
Traffic Manipulation	Python 2.7	Python 2.7

polling intervals between control-plane and data-plane also increase, which utilizes very large bandwidth.

It is also mandatory to maintain polling bounds given as LB_{min} and UB_{max} for each flow passing through switches, where stable flows are placed in higher polling intervals and lower polling intervals will be used for heavy, busy and unstable flows [25]. We manipulated the tunning frequency by sending FlowStatisticsRequest Messages for individual flows. Control-plane messages are not capable to capture acknowledged flow traffic statistics, instead, we gathered flow statistics by simple multiplication of sampling ratio with the number of bytes in each flow, addition to this we also utilized sampling based adaptive polling interval algorithm elaborated in Algorithm 1. The equation 8 depicts rate of change with unit value if the predicted status is very closer to the actual status. All other values must satisfy given condition of $0 \leq \beta \leq 1$, where 0 and 1 values represents minimum value of M and maximum value of M respectively. In the Table 4, we depicted feasible rules to manipulate $\Delta Time_{New}$ based on values of M, where M_{min} is set to 0.6 seconds and M_{max} is set to 1.4 seconds. The main reason to set these values is to acquire the best performance of switches polling query. Our adaptive polling-based sampling inside all switches mainly focuses on time-oriented configurations to maintain the network data sampling for higher accuracy. If the upper bound limit is set to a large value, then network data after sampling will not be up to date, majority of live network traffic will not be recorded. Similarly, if we reduce lower bound time to live (TTL) in the polling algorithm, then it also causes degrade the sampling accuracy . In our algorithm, we tried various upper bound and lower bound conditions based on network traffic in our test-bed and configurations mentioned in Table 4, these parameters of adaptive polling achieves best sampling performance in switches across the test best.

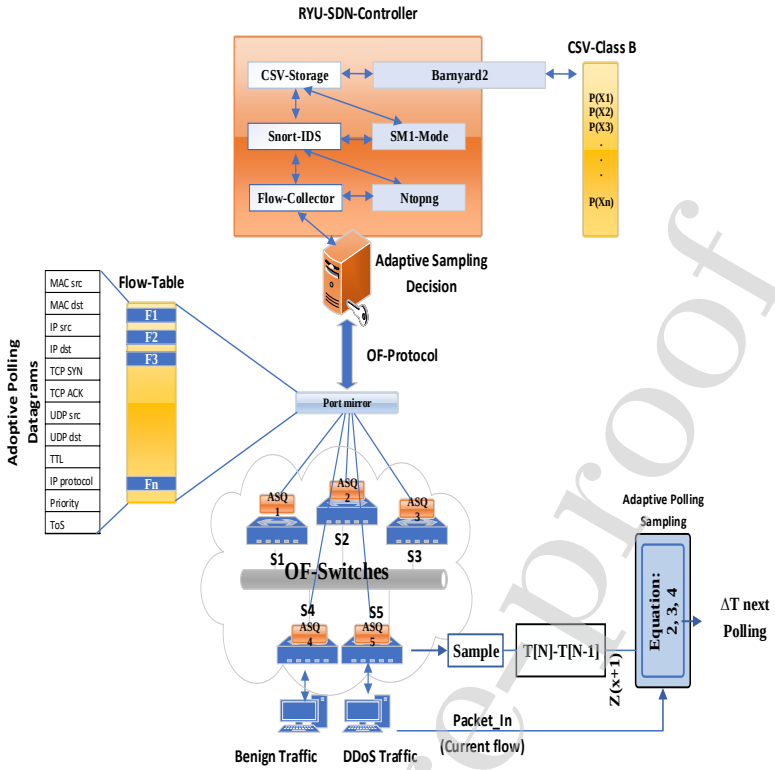


Figure 2: Data Acquisition with Adaptive Polling Interval Sampling.

Table 4
Rules to adjust intervals for Polling sampling.

Calculated M Values	New Sample Values
$M < M_{min}$	$\Delta Time_{New} = M \times \Delta Time_{curr}$
$M_{min} \leq M \leq M_{max}$	$\Delta Time_{New} = \Delta Time_{curr}$
$M_{max} < M$	$\Delta Time_{New} = \Delta Time_{curr} + 1_{sec}$
M not defined	$\Delta Time_{New} = 2_{sec} \times \Delta Time_{curr}$

4.4. Snort-IDS (Adaptive Polling Sampling)

In this module, we deployed Snort as NIDS with promiscuous enabled mode. The traffic from adaptive switches query with a sampling threshold is propagated to Snort detection engine to log data streams. Sampled traffic is visually analyzed and maintained with Ntopng tool [47]. This module uses two modes of packet logging with Snort IDS. SM1-mode is slightly different from the previous mode as shown in Table 5. SM1-mode is applied for malicious traffic collection and SM2-mode for benign traffic collection. This module uses Ntopng tool with OpenFlow enabled switches and software-based Ryu controller, which helps traffic engineer to predict and regulate the sampled traffic based on traffic behaviour and relevant features. In our proposed work, we created a SDN based application to collect sampled traffic with Snort signature based modes.

Table 5
Snort Traffic Acquiring Modes with AP sampling.

Description	Snort Two Modes	
	SM1 (Malicious)	SM2 (Benign)
TCP rules	15 signatures	8 signatures
UDP rules	10 signatures	6 signatures
ICMP rules	6 signatures	6signatures
CSV Database	Barnyard2	Barnyard2
Log file	snort.log.xx3	snort.log.xx4
Traffic Monitor	Ntopng	Ntopng
Traffic Manipulation	Python 2.7	Python 2.7

4.5. CSV-storage (sFlow and adaptive polling Sampling)

This module is very fundamental for classifying different feature extractions and collecting network traffic as a CSV file. In this module, we used Barnyard2 a well known MySQL database, which collaboratively works with Snort IDS. It is capable to reduce the workload of Snort as it writes Snort events in human readable modes either on console, text or CSV file. We utilized Barnyard2 to read all binary outputs from Snort into a MySQL database in the CSV form. In Barnyard2 database we deployed unified barnyard2 plug-in and CSV plug-in to acquire easily readable CSV file such as snort.sflow-log.xxx for benign and malicious traffic flow. However, data is gathered in CSV form in two different ways of SM1-mode and SM2-mode. Similar to sFlow this mod-

ule also uses Barnyard2 with unified plug-in and csv-plug-in for adaptive polling sampling, where Snort creates two separate files such as snort.asq_log.xxx for normal traffic and malicious traffic.

5. Traffic Analysis

Data acquisition with virtual machine (VM) is utilized in our work, where both benign data and malicious data is collected based on signature-based IDS with sampling approaches. Our research work mainly focuses on real-time network data acquisition. Our data acquisition methodology uses two types of sampling sFlow and adaptive polling interval sampling approaches. In data-plane, we deployed both sampling approaches individually and we gathered malicious and benign datasets with Snort SM1-mode and SM2-mode. We utilized two virtual machines, where VM1 is only used for data-plane and implementing sampling, VM2 machine is created with signature-based Snort IDS, Barnyard2 and central SDN Ryu controller. Both virtual machines communicated via OpenFlow 1.3 protocol and sFlow protocol, which is depicted in Figure 1 and Figure 2. To lower down the workload from detection engine of Snort, Barnyard2 is implemented that supports to capture maximum data with monitoring ability. Snort detection engine file is configured with specific rules to generate DDoS alerts.

In sFlow sampling we used 15, 10 and 6 rules for TCP, UDP and ICMP respectively in *snort.conf*, refers to Snort-mode (SM1) file for malicious dataset. Similarly, we utilized 8, 6 and 6 simple rules in *snort.conf* refers to Snort-mode (SM2) file for acquiring normal dataset as show in Table 3. In adaptive polling based sampling we also used 15, 10 and 6 rules for TCP, UDP and ICMP respectively in *snort.conf* refers to Snort-mode (SM1) file for the malicious dataset. Similarly, we utilized 8, 6 and 6 simple rules in *snort.conf* refers to Snort-mode (SM2) file for acquiring normal dataset detail is provided in Table 5.

Snort is capable to offer various functional outputs such as alert_syslog, alert_fast, alert_full, alert_unixsock, log_tcpdump and alert_csv, any of these standard Snort directives can be utilised in snort.conf file to achieve desirable network output packets file.

Our paper utilises the alert_csv output plugin in collaboration with SM1-mode and SM2-mode. We have utilised SM1-mode with 15-10-6 rules and SM2-mode with 8-6-6 rues. The main reason for using limited signatures is to write network alerts details in a csv file to process with DNN DDoS detection model. We did not use various Snort signature on snort.conf as it significantly reduced the quality of features extracted from maclious and benign traffic.

In SM1-mode, we configured all TCP, UDP and ICMP signatures with specific field values, such as port numbers, content numbers, threshold types and values, priority values, track by_src, track by_dst and classtype. These signatures values were filtering and classifying network data as malicious output. In SM2-mode we utilised only default signature values from Snort repository to record each and every packet, SM2-mode is used to collect only benign data. Further details of this implementation are also provided in Table 3.

Table 6
Common List of Extracted Headers for TCP, UDP and ICMP packets.

TCP	UDP	ICMP
Src-IP	Src-IP	Src-IP
Dst-IP	Dst-IP	Dst-IP
Src-Port	Src-Port	ICMP-code
Dst-Port	Dst-Port	ICMP-Type
protocol-Type	protocol-Type	protocol-Type
Packet-Length	Packet-Length	Packet-Length
TTL	TTL	TTL
SYN-ACK	Length	DF Flag
Window	Checksum	Timestamps

Table 7
Feature Extraction from TCP Flows.

1	No of packets in each incoming TCP-flows
2	No of packets in each outgoing TCP-flows
3	Change of packets in each incoming TCP-flows
4	Change of packets in each outgoing TCP-flows
5	No of distinctive Src-IP per incoming TCP-flows
6	Change of of distinctive Src-IP per incoming TCP-flows
7	Total Bytes per incoming TCP-flows
8	Total Bytes per outgoing TCP-flows
9	Symmetric windows length per incoming TCP-flows
10	Distinctive windows length per incoming TCP-flows
11	No of symmetric incoming TCP-flows
12	No of asymmetric incoming TCP-flows
13	No of symmetric TTL values per incoming TCP-flows
14	No of asymmetric TTL values per incoming TCP-flows
15	No of asymmetric src-ports per incoming TCP-flows
16	Change of asymmetric src-ports per incoming TCP-flows
17	No of asymmetric Dst-ports per incoming TCP-flows
18	Change of asymmetric Dst-ports per incoming TCP-flows
18	No of $Dst - ports \leq 1024$ per incoming TCP-flows

Our traffic collector section examines all incoming OpenFlow messages at controller, where signature-based Snort IDS and Barnyard2 extracts various header fields based on sampled flows to make CSV files. The flow is TCP and UDP protocol combination, which comprises similar values of the type of protocol, source IP addresses and destination IP addresses, source ports and destination ports. ICMP contains the majority of similar fields other than source and destination port numbers. We used Barnyard2 *alert_csv* plug-in by editing *barnyard.conf* with csv features such as *timestamp, msg, proto, src, srcport, dst, dstport, ethsrc, ethdst, ethlen, tcpflags, tcpseq, tcpack, tcplen, tcpwindow, ttl, tos, id, dgmlen, iplen, icmptype, icmpcode, icmpseq*.

However, as shown in the Table 6, the features are extracted using Python and literature survey, Python and SAE unsupervised machine learning to derive proposed feature set depicted in Table 7, for 22 TCP flows, Table 8, for 15 UDP flows and Table 9, for 10 ICMP flows.

Table 8
Feature Extraction from UDP Flows.

1	No of packets in each incoming UDP-flows
2	No of packets in each outgoing UDP-flows
3	Change of symmetric packets each incoming UDPflows
4	Change of asymmetric packets each incoming UDP-flows
5	No different Src-IP in each incoming UDP-flows
6	Total Bytes per incoming UDP-flows
7	Total Bytes per outgoing UDP-flows
8	Total packets per incoming UDP-flows
9	Total packets per outgoing UDP-flows
10	No different Src-ports in each incoming UDP-flows
11	No different Dst-ports in each incoming UDP-flows
12	No different $Dstports \leq 1024$ per incoming UDP-flows
13	No different $Dst - ports < 1024$ per incoming UDP flows
14	No of asymmetric TTL values per incoming UDP flows
15	No of symmetric TTL values per incoming UDP-flows

Table 9
Feature Extraction from ICMP Flows.

1	No of ICMP-flows in each incoming flows
2	No of ICMP-flows in each outgoing flows
3	No of symmetric incoming ICMP-flows
4	No of symmetric Src-IP per incoming ICMP-flows
5	Total Bytes per incoming ICMP-flows
6	Total Bytes per outgoing ICMP-flows
7	No of packets per incoming ICMP-flows
8	No of packets per outgoing ICMP-flows
9	Symmetric windows length per incoming TCP-flows
10	Different TTL values per incoming ICMP-flows

6. Evaluation

Our model mainly focuses to attack capture-failure rate with the false-negative rate. False-negative rate is highly used performance metrics of IDS systems, it is represented as when IDS systems fail to classify attacks that have been taken place on it. Our overall proposed model is based to improve the capture-failure rate and to provide a comparison between sampling approaches, which is another performance metric in this paper. The capture-failure rate implies as unawareness of the system to detect the attack on network infrastructure. This paper takes advantages of both sampling techniques with SDN flexibility such as sFlow sampling and adaptive polling sampling inside OVS switches with Ryu controller. In this work, the standard amount of sampled traffic flows are utilized with proposed sampling rather than deep packets sampling. In contrast, deep packets sampling utilizes many available network and processing resources. The detailed systematic overview of our proposed model is presented in this section.

6.1. Experimental Setup

Our experiment is performed into two different virtual machines, which is created with Intel (R) Xeon (R) X5560 CPU with 2.88 GHz processor and 16 GB RAM (DDR3 ECC-Registered Memory PC3-12800) running TensorFlow 1.4 on Ubuntu LTS 16.04-64 bit. VMware Player is used for creating both virtual machines such as VM1 with 192.168.232.x1 IP address and VM2 with 192.168.232.x2 IP address. The VM1 machine, uses network emulator tool such as Mininet tool for creating and customizing IoT network topology presented in Figure 3, sFlow sampling agents and adaptive polling intervals based sampling parameters are also implemented in Mininet based Open vSwitch (OVS). Snort-IDS as NIDS is utilized to actively collects sampled normal and malicious traffic streams, then Snort output plug-in creates csv data files with Barnyard2. In the same VM1, various malicious traffic of DDoS floods is emulated with Metasploit, hping3 and LOIC tool. However benign traffic is generated to evaluate our detection model. VM2 utilizes more tools and technologies, so more CPU processing and memory are assigned for flexible performance. In VM2, the model utilizes central SDN Ryu controller, and the SAE a deep neural network model for intrusion detection, which classifies traffic features and detects malicious traffic, this will be further discussed in results section with more details. Figure 4, represents the simplified flow diagram of our proposed design.

SDN Ryu controller centrally controls VM2 to install sampling policies and other network manipulations via Rest APIs configuration. VM1 is developed as data-plane and VM2 for control-plane, where VM1 communicates with VM2 by OpenFlow 1.3 protocol. The SDN Ryu controller manipulates and manages OpenFlow OVS switches in data-plane. The ovs-ofctl utility is used to insert new policies into switches table. The ovs-ofctl programme is used for OpenFlow switch administration and monitoring purpose, this programme is configured with north-bound APIs. SDN Ryu controller and Snort are integrated with each other, where Snort utilizes promiscuous enabled mode at eth0 interface which helps to collect all network traffic and propagate these packets to OpenFlow switch via port mirroring. On eth1, Snort and Ryu socket with IP 192.168.232.x1 is configured for collecting packets remotely. Snort continuously propagates data towards control-plane with 192.168.232.x2 IP address, this enables detection algorithms to work with the collaboration of Snort.

In VM2, OVS is remotely managed by Ryu controller inside Mininet simulator. The Snort switch ($switch\_snort.py$) application is programmed on top of Ryu controller, this supports Layer L2 switch coding and redirects all traffic via OpenFlow switches with promiscuous enabled mode, traffic is redirected between one of the Snort port and Ryu (Unix Socket) port. The Snort supports two integration option with Ryu controller, the first option is very basic only suitable for demonstration purpose. On the other hand, we utilizing the second option, where Snort is implemented with a remote machine. SDN Ryu controller receives alerts from Snort with *unixsock* network socket. When the Snort uses $unixsock = false$, then it receives all network packets for forwarding to Barnyard2 log file. This is the main reason to manage data-plan of network and hosts packets within main SDN controller. The tools utilized in our work are presented in Table 11.

In VM2, Tcpreplay tool is also implemented for the purpose of classifying and analyzing benign and malicious traffic individually. Addition to this, Python is used for manipulating and saving computed features from each packet acquisition intervals. Finally, the dataset file is categorized into

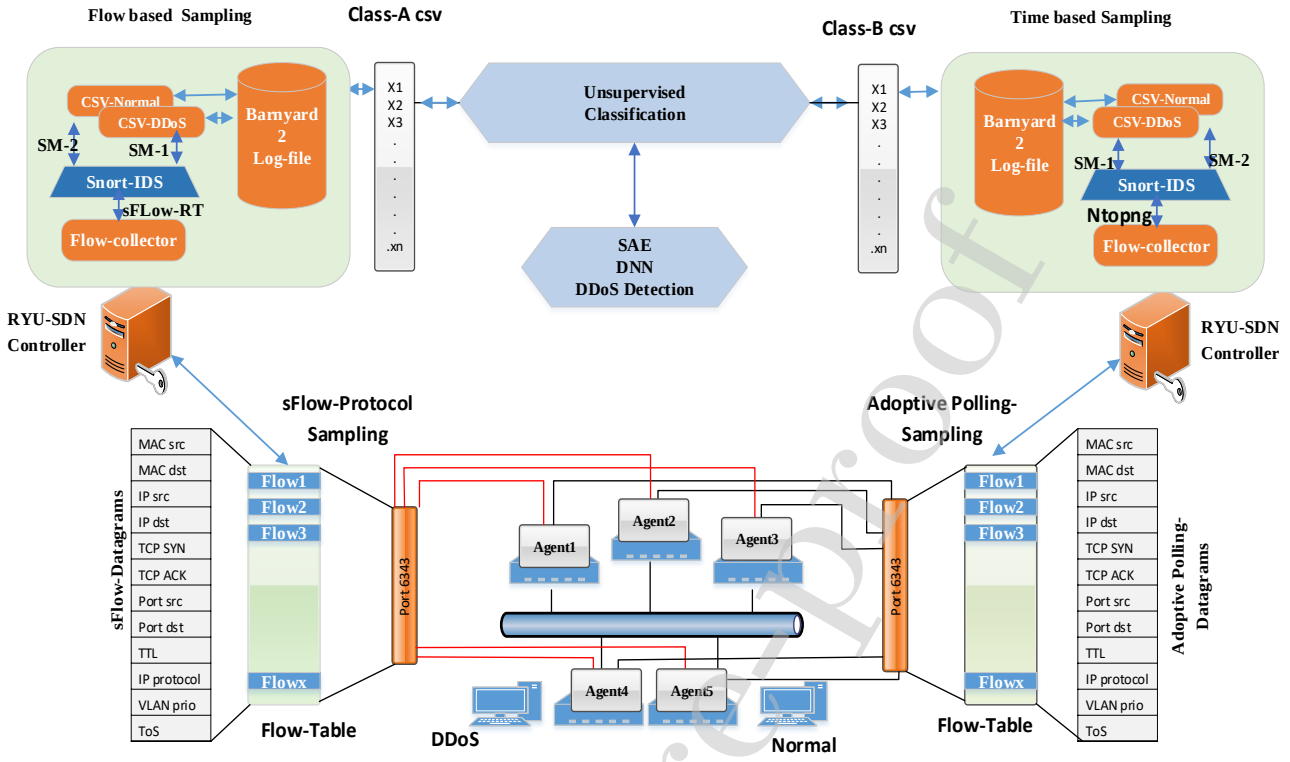


Figure 3: Proposed System in SDN.

training and testing datasets. Table 10 depicts the dataset distribution, all feature values are positive non-zero numbers. This model utilizes unity based feature scaling, Min-Max normalization approach to breaks all real-valued into 0 and 1 ranges as shown in equation 9, where model uses 60 seconds time (start to end) fitting slot to extract features. After data normalization, proposed work collaboratively works with widely used DNN model of SAE to detect malicious and normal activities.

$$P(X)_{normalized(0\ to\ 1)} = \frac{P(X)_i - P(X)_{min}}{P(X)_{max} - P(X)_{min}} \tag{9}$$

$$P(X)_{max} = maximum\ value\ range\ in\ P(X)_i$$
$$P(X)_{min} = minimum\ value\ range\ in\ P(X)_i$$
$$P(X)_i = total\ set\ of\ observed\ values$$

6.2. Results

This section mainly focuses on performance comparison of detection rate of DDoS within sFlow and adaptive polling based sampling approaches, the overall performance of the proposed mechanism is evaluated on the datasets presented in Table 10. As discussed in Section IV, traffic sampling can either be deployed as packet-oriented or time oriented. The control-plane uses SAE detection model with sampled and extracted features of data-plane and it also comprises of major sampling policies, flow policies, and OVS insertion rules.

Table 10
Representation of normal and attack records for training and testing.

CSV Traffic Class 1	sFlow Traffic	
	Training	Testing
Normal	45,263	12,528
$All_{attacks}$	34,193	7,347
CSV Traffic Class 2	**Adaptive Polling Sampling Traffic**	
	Training	**Testing**
Normal	43,895	11,259
$All_{attacks}$	35, 097	8,564

In order to get feasible classification results, this work uses SAE, a simple DNN model with extracted features. Firstly, features based input vectors acquired from sFlow and adaptive polling based sampling datasets, then SAE applied to classify benign and malicious traffic. However, this type of dataset comprises a higher level of unbalanced features, and in the real network there are very small portions of malicious traffic as compared to emulated malicious traffic. Most of the learning models create alarming situations for such reasons. Due to this fact, huge numbers of neural network models classify all traffic streams as benign and malicious as traffic noises. To achieve higher accuracy, a vast variety of approaches can be deployed to overcome this situation [27]. As a solution to this case, we utilize weighted loss functions to balance unbalanced feature frequencies, this work also utilizes selected feasible metrics such as *precision, recall and*

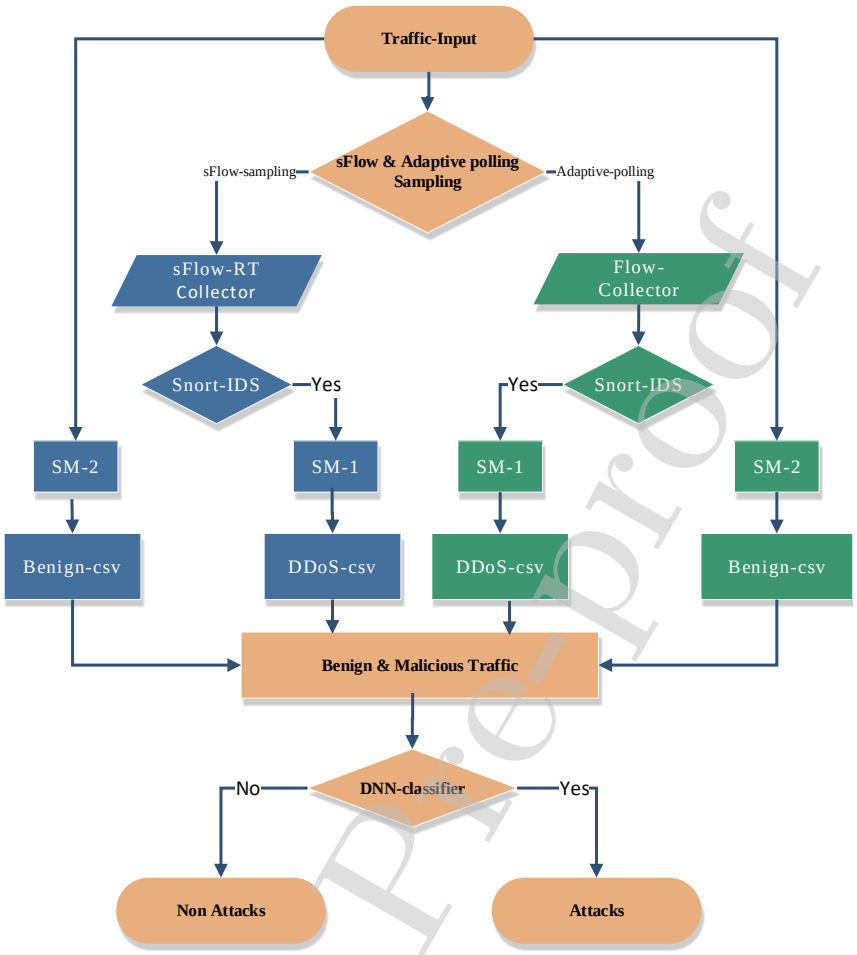


Figure 4: Flow diagram of proposed work.

Table 11
Test-bed tools and technologies.

Technologies	Name (Version)	Description
Simulator	Mininet-2.2.1 [36]	Popular network emulator to create virtual switches, hosts and networks links. A realistic test-bed created on VM1 virtual machine, running real kernel, ovs switches, and application code. Data-plane and control-plane communicate with OpenFlow 1.3 protocol.
SDN-controller	Ryu-4.27[53]	Ryu is component based SDN development framework with well defined API to cater flexible and innovative network-management and control-applications. It is open source, works with OpenFlow, Netconf, OF-config etc. Ryu code is written with Python which is available under Apache 2.0 license
IDS-Sensor	Snort[57]	Open source, widely used intrusion detection system with various active detection and logging facility.
Traffic-Analyzer	Tcpreplay	Open source UNIX based utility to edit and replaying network traffic, helps to rewrite and classify traffic at various layers on client and server side. Specially works with IDS/IPS systems
Attacking Tools	hping3[23] LOIC [1]	Well known freely available application, mainly used for netwrok stress testing with DoS/DDoS attacks emulation. These toos can emulate huge TCP, UDP, HTTP-GET floods to specific target via IP address of target machines.
Development-kit	Python-2.4	Python a well known scripting language is utilized for script development specially for routing flow management, traffic sampling and ovs-switches.

F1 score on dataset specified in Table 10. These metrics provide useful information about the goodness of the detection rate of the proposed model. These parameters are depicted with the help of a confusion matrix, and its entries are provided as below:

1. **True Positive (TP)** - these values are correctly identified attacks records.
2. **True Negative (TN)** - these values are correctly identified non attack values.
3. **False Positive (FP)** - output values are incorrectly predicted attack records.
4. **False Negative (FN)** - output values are incorrectly identified non attacks
5. **Precision (P)**: - calculates proportion of predicted positive cases (real attacks) with correct values:

$$P = \frac{TP}{TP + FP} \tag{10}$$

6. **Recall (R)**: calculates percentage of predicted NIDS malicious activities with all available attacks. Higher value of R is very important:

$$R = \frac{TP}{TP + FN} \tag{11}$$

7. **F-measure (F)**: The NIDS model test accuracy is calculated by utilizing harmonic mean with both of precision (P) and the recall (R) values, where higher F score is considered feasible:

$$F = \frac{2}{\frac{1}{P+R}} \tag{12}$$

Our study uses the two different dataset portions, based on sFlow and adaptive polling sampling as discussed before for training and testing, each dataset is divided into training and test sets followed by 80% of training and cross-validation and 20% of test sets. This work uses a combination of benign and normal datasets for training and cross-validation purpose, to make detection approach more realistic and enabling generalization for detecting new attack behaviours, which is a very crucial aspect for network security. Considering all these factors, the proposed solution employs a deep learning model with the appropriate classification. Where the model achieves great results as feasible evidence of our proposed study. For this mandatory purpose, we set SAE domain with only three hidden-layers with encoding/decoding elements of 10, 5, 2, all three layers processed following by descending order of 22-10-5-2-1, 22-10-5-1, 22-10-1. The hyper parameters and tested numeric values are depicted in Table 12.

This paper utilizes the two-class classification model for normal and attack class including TCP, UDP and ICMP traffic streams. For better comparison, the SAE detection model

Table 12
Proposed model configuration values.

Hyper parameters	Model values
Total hidden-layers	1, 2, 3
Hidden-layers size	10, 5, 2
Learning-rate	0.1, 0.01, 0.001
Activation-function	Sigmoid
Dropout	0.75, 0.50
Batch-size	100, 50, 50

is employed individually for sFlow and adaptive polling based datasets. After that the evaluation of the precision, recall, F-measure, accuracy and false-positive values is carried out for both traffic classes, these values are derived from confusion matrix shown in Figure 5. Value of comparison is depicted in Figure 6. Figure 5 presents evaluation results of sFlow and adaptive polling based sampling approaches, where, classification model evaluated with two class model separately for sFlow and adaptive polling sampling considering TCP, UDP and ICMP DDoS attacks into single class against normal class traffic. Figure 5 (a) confusion matrix for sFlow based two-class classification achieves nearly 95% of TP detection rate with only 4% of FP rate, on the other hand in Figure 5 (b), adaptive polling based sampling detection rate of TP rate was around 92%, whereas FP rate was higher than 8%. Figure 6 depicts detailed comparison between both sampling approaches by using standard classifier parameters as it can be observed from Figure 6, DNN model with sFlow class achieved accuracy of 91% with greater *F_measure* values of 88.10%. Similarly DNN model with adaptive polling based sampling achieved accuracy of 89% with 85% of *F_measure* values. Precision values for sFlow and adaptive polling based sampling was 95% and 92% respectively. Recall values were also nearly 5% higher for sFlow as compared to adaptive polling. Table 13 depicts the model summary of evaluation in term of Accuracy, Precision, Recall, F-sores, TP-rate, TN-rate, FP-rate and FN-rate.

Figure 7 depicts two classes ROC curve to provide overall DNN model detection performance, with sFlow based implementation model achieves greater sensitivity probability values of 94% for positive model test outcomes, however, this implementation model receives less than 4% of negative outcomes of DNN model test. In contrast, the polling based implementation model receives positivity of DNN model at 92% values of sensitivity with nearly 8% of negativity values of specificity. Moreover, the overall performance of sFlow based implementation was satisfying with our proposed DNN model in testbed in various network terms such as packets flows, network, CPU and memory and network overhead.

6.3. System Performance Evaluation

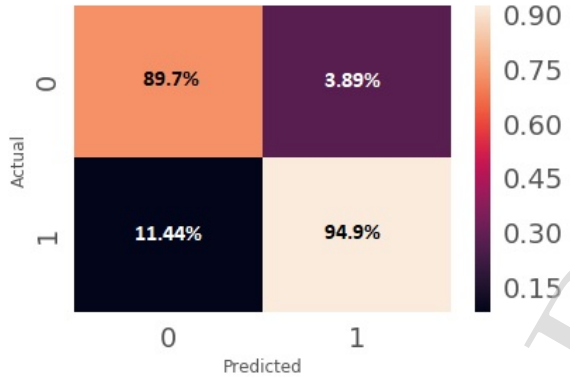
This section depicts the comparison of average CPU utilization for sFlow based and adaptive polling based sampling approaches between data-plane to control-plane locations, which is considered as highly influencing factor for detection model. Figure 8 (a) presents the different CPU usages

Table 13
Summary of two class model evaluation for sFlow and adaptive polling sampling.

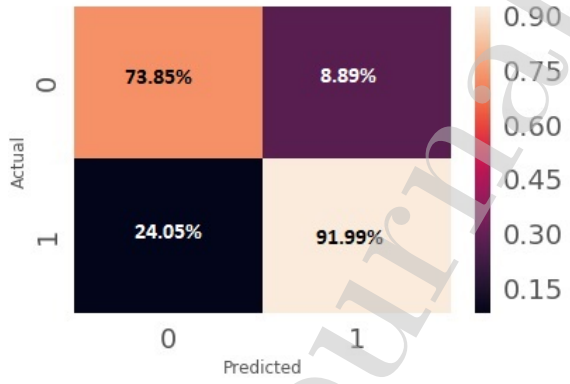
Sampling types	TP	TN	FP	FN	Accuracy-value	Precision-value	R-value	F-value
sFlow	95%	89%	4%	11.44%	91%	95%	83%	88.10%
Adaptive Polling	92%	74%	8.89%	24%	89%	92%	78%	85%

Table 14
Summary of total CPU utilization of controller.

CPU Utilization(%)								
sFlow Sampling					Adaptive Polling Sampling			
Time(sec)	CPU(TCP)	CPU(UDP)	CPU(ICMP)	CPU(All)	CPU(TCP)	CPU(UDP)	CPU(ICMP)	CPU(All)
20	36	22	21	57	30	22	29	80
40	39	33	17	60	39	29	31	63
60	29	30	18	42	25	56	21	65
80	38	40	16	50	44	21	33	69
100	22	31	23	53	33	50	32	72
120	27	57	28	70	49	37	37	64
140	55	13	27	74	11	10	18	56



(a) sFlow Sampling.



(b) Adaptive Polling based Sampling.

Figure 5: Confusion Matrix of 2 Class Classification.

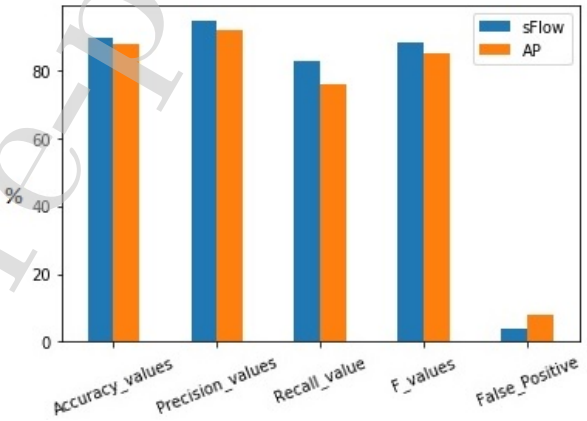


Figure 6: Accuracy, Precision, Recall, F-sores and FP-rate for two class model.

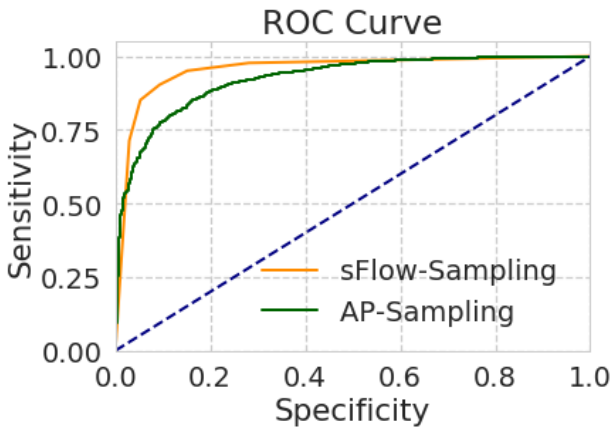


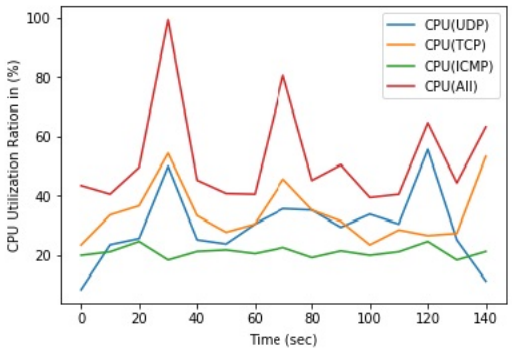
Figure 7: ROC Curve of 2 Class Classification.

such as CPU(TCP), CPU(UDP), CPU(ICMP), CPU(All).

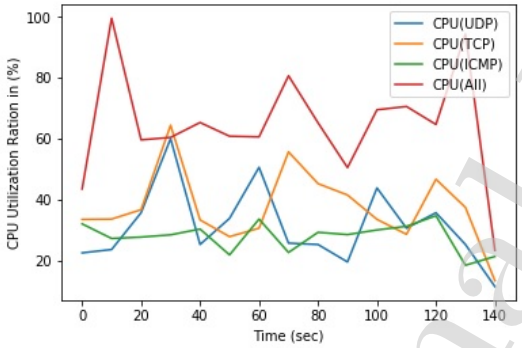
The summary of CPU utilisation for sFlow and adaptive polling sampling with time is depicted in Table 14. The proposed model receives TCP, UDP and ICMP attack with 100 Kb/s to 600 Kb/s. Overall 290 M Bytes of total malicious traffic injected to observe CPU utilization individually only at control-plane after sampling, from Figure 8 (a), it can be observed that CPU(UDP) only uses 29% of CPU utilization with 2164 encoding and decoding threads instructions. The CPU(TCP) utilizes only 38% CPU with 2479 CPU threads, In ICMP the CPU is constantly used at 26% with 2011 threads. Due to sFlow sampling, it can be observed that overall CPU utilization reduced from 90% to

Table 15
Summary of total network load of controller.

Controller Network Load (KB/s)				
sFlow Sampling			Adaptive Polling Sampling	
Time(sec)	Control-plane (packets)	Data-plane (packets)	Control-plane (packets)	Data-plane (packets)
20	130	330	110	330
40	357	1400	250	1400
60	470	1078	430	1078
80	366	7600	380	7600
100	378	600	330	600
120	358	650	311	650
140	578	702	230	702

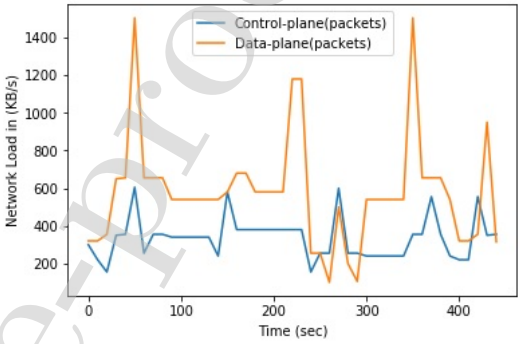


(a) sFlow Sampling.

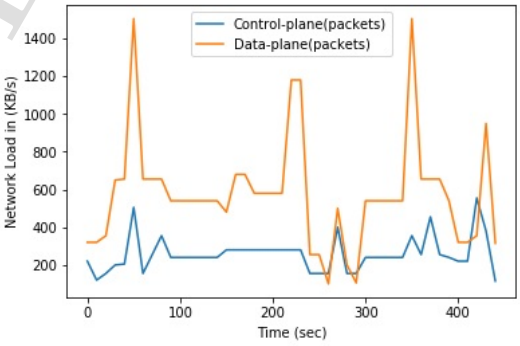


(b) Adaptive Polling based Sampling.

Figure 8: CPU Utilization of Controller over time.



(a) sFlow Sampling.



(b) Adaptive Polling based Sampling.

Figure 9: Controller Network Load over time.

52%. Similarly, in Figure 8 (b), based on adaptive polling sampling, CPU(TCP) utilization increased from 38% to 42% with threads execution of 2659. CPU (UDP) utilization also raised to 37% with 2551 threads. CPU utilization for ICMP stood at 6% higher than sFlow with some threads increment. However, in polling based sampling overall CPU usage was also higher as 74.25% during TCP, UDP, ICMP malicious data flow due to the nature of being calculating flows and packets equation individually in various switches.

Figure 9 depicts the network load of sFlow sampling and adaptive polling based sampling between two virtual machines VM1 data-plane and VM2 control-plane. In VM1 data-plane(packets) represents injected traffic flows before sampling, and control-plane(packets) represents flows received by the controller after sampling. The network overhead calculated with malicious packets of TCP, UDP, ICMP flooding attacks. Malicious traffic emulation rate was gradually changed from 150 KB/s to 1500 KB/s, which was kept the common flow for sFlow and adaptive based polling sampling to calculate network overhead, where Ntopng tool is used to analyze network manipulating facts. Summary of the controller network load with time is also presented in Table 15. As shown in Figure 9 (a), once controller connected to sFlow agent enabled switches then it received accumulated average packets flows of 340 KB/s after sFlow sampling, due to link utilization and network queue packet flow was fluctuating nearly 500 KB/s after every 40 seconds. The controller received total control-plane (packets) 192 MB out of 290 MB. Similarly, as we can observe from Figure 9 (b), the controller was receiving accumulated average packets at the rate of 244KB/s, Due to polling-based sampling, controller

received total 140MB out of 290 MB. Moreover, adaptive polling based sampling measures higher network load due to the fact that this sampling requires individual calculations and queering FlowStats Request inside all switches flow table, it utilizes more CPU which make flow slower to reach at controller detection model.

7. Discussion

Our proposed system is based on IoT nodes sampling which effectively works in collaboration of SAE deep neural network model. It collects flows from data-plane via packet based sampling (sFlow) and time-based (Adaptive Polling) sampling respectively. Overall our model has clear advantage of detecting the enormous amount of malicious traffic either with packet-based or flow based sampling approaches which reduce the Snort and deep learning models overhead processing, as intrusion detection systems are limited with processing and detection capabilities. Similarly, DNN models are also demanding more training time and complex configurations during the event of heavy flows. This enabled us to parse all crucial malicious and benign traffic streams to classify and label correctly. In case of mitigation or blocking any switches or hosts unwanted incoming flow would be more accurate and reliable task for SDN controller with simple northbound APIs. In collaboration, an SAE model individually performs on the attack and normal classes, which successfully achieve accuracy nearly 99% on normal traffic to qualify our proposed model parameters before our proposed model test.

Most of the common features from both of proposed sampling extracted with 22 input vectors shown in Table 7, 8, 9. The proposed solution derives common features following the literature of [45] then merge all of the testbed features with only 22 feature vectors class. As observed from experimental results, sFlow achieved a higher detection rate of TP = 95%, TN = 90% with less than 4% of FP rate, whereas polling based implementation detection was around 5% less as compared to sFlow. Overall experiments represent that sFlow based implementation in our proposed work has significantly higher detection accuracy with minimal CPU and network load. There are some important factors that support this point, such as sFlow is flow oriented sampling technology, which utilizes sFlow agents in OVS switches with a sampling rate of 1/28, 1/ 64, 1/128, 1/256, 1/512 for s1 to s5 respectively. In this way, accurate sampled traffic was flexibly flowing to Snort IDS at the controller. The controller received around 192 M Bytes of total packets within 20 minutes of packets injection, the controller used 50% of total CPU overhead. On contrary the polling based implementation, we can observe that controller only receives 140 M Bytes within 20 minutes of packets injection, which is at least 27% fewer packets than sFlow at detection engine of Snort at the controller. Due to the fact of adaptive polling uses higher resource consumption to manipulate each packet inside switches and constantly sends each packet once match polling query. In this way, external traffic collectors have to wait for the next packets. Moreover, this implementation used 74% of total CPU, which is significantly higher for 140 M bytes of network packets as in real-time existing networks there large number of packets flows passing through ingress and egress ports of autonomous and service providers network.

8. Conclusion and Future Directions

This paper presents co-detection model of deep learning with Snort IDS, to detect unwanted IoT nodes DDoS traffic with the help of sFlow and polling based traffic sampling at data-plane. This work analyzes network overhead between the SDN controller and data-plane with each sampling implementation to improve detection of IDS and DNN model capabilities. sFlow as flow oriented sampling utilized 50% of CPU with propagation rate of 340 KB/s packets towards detection unit of the controller, whereas adaptive polling based uses more than 70% total machine CPU at parsing rate of around 240 KB/s. Moreover, sFlow measures very low CPU and network overhead without sacrificing detection accuracy of deep learning model which is based on real-time detection of traffic with Snort IDS. DDoS detection accuracy trade-off between sFlow and adaptive polling based sampling was achieved 95% with less than 4% FP rate and 92% with an FP rate of 8% respectively. However, sFlow consistently stood superior for proposed DDoS detection model in term of accuracy, low CPU, and network overhead.

These adopted sampling techniques are mandatory to support the future IoT application requirements, which can provide flexible and efficient data handling for classifying DDoS and benign traffic by reducing data for detection model of IoT networks. For future improvement, there are several directions being considered. First, to implement intelligent periodic polling based sampling only at SDN-enabled edge switches with real-time traffic streams that can help to lower down the crucial overhead. Secondly, it is very vital to train unsupervised deep learning model with two individual training of rule-based and signature-based real-time network datagrams, which could help to maximize DDoS detection accuracy as a whole, which is suitable for small to larger networks.

Compliance with Ethical Standards

This research has been produced as a part of PhD work, no funding is sought for this research by the authors.

Conflict of Interest

Raja Majid Ali Ujjan, Zeeshan Pervez, and Keshav Dahal have declared that they have no conflict of interest.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] abatishchev, 2019. Loic a network stress testing application. URL: https://sourceforge.net/projects/loic/.

[2] Ahmed, M.E., Kim, H., 2017. Ddos attack mitigation in internet of things using software defined networking, in: 2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService), pp. 271–276. doi:10.1109/BigDataService.2017.41.

[3] Alanazi, S., Al-Muhtadi, J., Derhab, A., Saleem, K., AlRomi, A.N., Alholaibah, H.S., Rodrigues, J.J.P.C., 2015. On resilience of wireless mesh routing protocol against dos attacks in iot-based ambient assisted living applications, in: 2015 17th International Conference on E-health Networking, Application Services (HealthCom), pp. 205–210. doi:10.1109/HealthCom.2015.7454499.

[4] Androulidakis, G., Papavassiliou, S., 2008a. Improving network anomaly detection via selective flow-based sampling. IET Communications 2, 399–409. doi:10.1049/iet-com:20070231.

[5] Androulidakis, G., Papavassiliou, S., 2008b. Improving network anomaly detection via selective flow-based sampling. IET Communications 2, 399–409.

[6] Braga, R., Mota, E., Passito, A., 2010a. Lightweight ddos flooding attack detection using nox/openflow, in: IEEE Local Computer Network Conference, pp. 408–415. doi:10.1109/LCN.2010.5735752.

[7] Braga, Braga, R., Mota, Mota, E., Passito, Passito, A., 2010b. Lightweight ddos flooding attack detection using nox/openflow, in: Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks, IEEE Computer Society, Washington, DC, USA. pp. 408–415. URL: http://dx.doi.org/10.1109/LCN.2010.5735752, doi:10.1109/LCN.2010.5735752.

[8] Bu, C., Wang, X., Huang, M., Li, K., 2018. Sdnfv-based dynamic network function deployment: Model and mechanism. IEEE Communications Letters 22, 93–96. doi:10.1109/LCOMM.2017.2654443.

[9] Carlos, A.B.M.; Rothenberg, C.M.F., 2010. In-packet bloom filter based data center networking with distributed openflow controllers. In Proceedings of the 2010 IEEE GLOBECOM Workshops (GC Wkshps), , 584 –588.

[10] Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W., Peng, J., 2018. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud, in: 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 251–256. doi:10.1109/BigComp.2018.00044.

[11] Cisco, 2019. Cisco visual networking index predicts near-tripling of ip trac by 2020. URL: https://newsroom.cisco.com/press-release-content?articleId=1771211.

[12] Corp, I., 2019. Inmon corp. URL: https://sflow-rt.com/.

[13] Dong, P., Du, X., Zhang, H., Xu, T., 2016. A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows, in: 2016 IEEE International Conference on Communications (ICC), pp. 1–6. doi:10.1109/ICC.2016.7510992.

[14] Erfani, S.M., Rajasegarar, S., Karunasekera, S., Leckie, C., 2016. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. Pattern Recognition 58, 121–134.

[15] Flauzac, O., GonzÃąlez, C., Hachani, A., Nolot, F., 2015. Sdn based architecture for iot and improvement of the security, in: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, pp. 688–693. doi:10.1109/WAINA.2015.110.

[16] Giegel, J., 2018. inmon. URL: http://www.inmon.com.

[17] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V., 2014. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. Comput. Netw. 62, 122–136. URL: http://dx.doi.org/10.1016/j.bjp.2013.10.014, doi:10.1016/j.bjp.2013.10.014.

[18] Ha, T., Kim, S., An, N., Narantuya, J., Jeong, C., Kim, J., Lim, H., 2016. Suspicious traffic sampling for intrusion detection in software-defined networks. Comput. Netw. 109, 172–182. URL: https://doi.org/10.1016/j.comnet.2016.05.019, doi:10.1016/j.comnet.2016.05.019.

[19] Jacobson, V., 1995. Congestion avoidance and control. SIGCOMM Comput. Commun. Rev. 25, 157–187. URL: http://doi.acm.org/10.1145/205447.205462, doi:10.1145/205447.205462.

[20] Kacha, C.C., Shevade, K.A., Raghuwanshi, D.K.S., 2013a. Improved snort intrusion detec- tion system using modified pattern matching technique , 81–88.

[21] Kacha, C.C., Shevade, K.A., Raghuwanshi, D.K.S., 2013b. Improved snort intrusion detection system using modified pattern matching technique.

[22] Kain, E., 2018. 'world of warcraft: Legion' goes down as blizzard servers hit with ddos. URL: https://www.forbes.com/consent/?toURL=https://www.forbes.com/sites/erikkain/2016/09/01/world-of-warcraft-legion-goes-down-as-blizzard-servers-hit-with-ddos/.

[23] Kalitool, 2019. hping3 package description. URL: https://tools.kali.org/information-gathering/hping3.

[24] Kandoi, R., Antikainen, M., 2015. Denial-of-service attacks in openflow sdn networks, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 1322–1326. doi:10.1109/INM.2015.7140489.

[25] Kaspersky, 2019. Kaspersky ddos intelligence report for q1 2016. URL: https://securelist.com/kaspersky-ddos-intelligence-report-for-q1-2016/74550/.

[26] Kawahara, R., Mori, T., Kamiyama, N., Harada, S., Asano, S., 2007. A study on detecting network anomalies using sampled flow statistics , 81.

[27] Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al., 2006. Handling imbalanced datasets: A review. GESTS International Transactions on Computer Science and Engineering 30, 25–36.

[28] Kreutz, D., Ramos, F.M., Verissimo, P., 2013. Towards secure and dependable software-defined networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ACM, New York, NY, USA. pp. 55–60. URL: http://doi.acm.org/10.1145/2491185.2491199, doi:10.1145/2491185.2491199.

[29] Kurose, J.F., Ross, K.W., 2009. Computer Networking: A Top-Down Approach. 5th ed., Addison-Wesley Publishing Company, USA.

[30] Lindqvist, U., Neumann, P.G., 2017. The future of the internet of things. Commun. ACM 60, 26–30. URL: http://doi.acm.org/10.1145/3029589, doi:10.1145/3029589.

[31] Ma, H., Liu, L., Zhou, A., Zhao, D., 2016. On networking of internet of things: Explorations and challenges. IEEE Internet of Things Journal 3, 441–452. doi:10.1109/JIOT.2015.2493082.

[32] Mai, J., Sridharan, A., Chuah, C., Zang, H., Ye, T., 2006a. Impact of packet sampling on portscan detection. IEEE Journal on Selected Areas in Communications 24, 2285–2298. doi:10.1109/JSAC.2006.884027.

[33] Mai, J., Sridharan, A., Chuah, C.N., Zang, H., Ye, T., 2006b. Impact of packet sampling on portscan detection. IEEE J.Sel. A. Commun. 24, 2285–2298. URL: https://doi.org/10.1109/JSAC.2006.884027, doi:10.1109/JSAC.2006.884027.

[34] Mehdi, S.A., Khalid, J., Khayam, S.A., 2011. Revisiting traffic anomaly detection using software defined networking, in: Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, Springer-Verlag, Berlin, Heidelberg. pp. 161–180. URL: http://dx.doi.org/10.1007/978-3-642-23644-0_9, doi:10.1007/978-3-642-23644-0_9.

[35] Metasploit, 2018. Penetration testing software, pen testing security. URL: https://www.metasploit.com/.

[36] Mininet, 2019. An instant virtual network on your laptop (or other pc). URL: http://mininet.org/.

[37] Mohammadi, R., Javidan, R., Conti, M., 2017. Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. IEEE Transactions on Network and Service Management 14, 487–497. doi:10.1109/TNSM.2017.2701549.

[38] Mousavi, S.M., St-Hilaire, M., 2015. Early detection of ddos attacks against sdn controllers, in: 2015 International Conference on Computing, Networking and Communications (ICNC), pp. 77–81. doi:10.1109/ICCNC.2015.7069319.

[39] Nanda, S., Zafari, F., DeCusatis, C., Wedaa, E., Yang, B., 2016. Pre-

dicting network attack patterns in sdn using machine learning approach, in: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 167–172. doi:10.1109/NFV-SDN.2016.7919493.

[40] Networks:, J., 2019. Juniper networks junos 7.2 software documentation. URL: https://www.juniper.net/documentation/product/en_US/junos-os.

[41] Neumann, P.G., 2016. Risks of automation: A cautionary total-system perspective of our cyberfuture. Commun. ACM 59, 26–30. URL: http://doi.acm.org/10.1145/2988445, doi:10.1145/2988445.

[42] Newman, L., 2019. What we know about fridayâĂŹs massive east coast internet outage. URL: https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn.

[43] Ng, A., 2011. Sparse autoencoder. CS294A Lecture notes.

[44] Nguyen, T.H., Yoo, M., 2017. A hybrid prevention method for eavesdropping attack by link spoofing in software-defined internet of things controllers. International Journal of Distributed Sensor Networks 13, 1550147717739157. URL: https://doi.org/10.1177/1550147717739157, doi:10.1177/1550147717739157, arXiv:https://doi.org/10.1177/1550147717739157.

[45] Niyaz, Q., Sun, W., Javaid, A.Y., 2017a. A deep learning based ddos detection system in software-defined networking (sdn). ICST Trans. Security Safety 4, e2.

[46] Niyaz, Q., Sun, W., Javaid, A.Y., 2017b. A deep learning based ddos detection system in software-defined networking (sdn). EAI Endorsed Transactions on Security and Safety 4. doi:10.4108/eai.28-12-2017.153515.

[47] ntop, 2019. How enable dpi-based traffic management in pfsense using nedge. URL: https://www.ntop.org/ntopng/how-to-setup-nedge-to-support-pfsense-operation/.

[48] Phaal, P., 2002. Packet sampling basics. URL: http://www.sfow.org/.

[49] Phaal, P., Panchen, S., Mckee, N., 2001. Inmon corporations sflow: A method for monitoring traffic in switched and routed networks doi:10.17487/rfc3176.

[50] Prokhorenko, V., Choo, K.K.R., Ashman, H., 2016. Web application protection techniques: A taxonomy. Journal of Network and Computer Applications 60, 95 – 112. URL: http://www.sciencedirect.com/science/article/pii/S1084804515002908, doi:https://doi.org/10.1016/j.jnca.2015.11.017.

[51] Rawat, D.B., Reddy, S.R., 2017. Software defined networking architecture, security and energy efficiency: A survey. IEEE Communications Surveys Tutorials 19, 325–346. doi:10.1109/COMST.2016.2618874.

[52] Restflow, 2018. Restflow. Https://github.com/restflow-org/restflow/wiki.

[53] Ryu, 2018. Ryu. available online:. Https://osrg.github.io/ryu/.

[54] Sahoo, K.S., Tiwary, M., Sahoo, B., 2018. Detection of high rate ddos attack from flash events using information metrics in software defined networks, in: 2018 10th International Conference on Communication Systems Networks (COMSNETS), pp. 421–424. doi:10.1109/COMSNETS.2018.8328233.

[55] Shah, S.A.R., Issac, B., 2018. Performance comparison of intrusion detection systems and application of machine learning to snort system. Future Generation Computer Systems 80, 157 – 170. URL: http://www.sciencedirect.com/science/article/pii/S0167739X17323178, doi:https://doi.org/10.1016/j.future.2017.10.016.

[56] Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013. Avant-guard: Scalable and vigilant switch flow management in software-defined networks, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, New York, NY, USA. pp. 413–424. URL: http://doi.acm.org/10.1145/2508859.2516684, doi:10.1145/2508859.2516684.

[57] Snort, 2019. Snort-intrusion detection system and prevention. URL: https://www.snort.org/.

[58] Sonar, K., Upadhyay, H., 2014. A Survey : DDOS Attack on Internet of Things 10, 58–63.

[59] Tran, Q.A., Jiang, F., Hu, J., 2012. A real-time netflow-based intrusion detection system with improved bbnn and high-frequency field programmable gate arrays. 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications , 201–208.

[60] Wang, R., Butnariu, D., Rexford, J., 2011. Openflow-based server load balancing gone wild, in: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, USENIX Association, Berkeley, CA, USA. pp. 12–12. URL: http://dl.acm.org/citation.cfm?id=1972422.1972438.

[61] Wang, R., Jia, Z., Ju, L., 2015. An entropy-based distributed ddos detection mechanism in software-defined networking, in: 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 310–317. doi:10.1109/Trustcom.2015.389.

[62] Yan, Q., Yu, F.R., Gong, Q., Li, J., 2016. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE Communications Surveys Tutorials 18, 602–622. doi:10.1109/COMST.2015.2487361.

[63] Yang, S., Kim, Y., Kim, H., Yang, S., Lim, S., 2015. Controller scheduling for continued sdn operation under ddos attacks 51.

[64] Zhang, L., Yang, K.U.N., 2018. A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework 6.

[65] Zhou, L., Liao, M., Yuan, C., Sheng, Z., Zhang, H., 2015. Ddos attack detection using packet size interval, in: 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015), pp. 1–7. doi:10.1049/cp.2015.0754.

The authors declares there is no conflict of interest regarding the publication of this journal paper.

Raja Majid Ali Ujjan is a PhD candidate at the University of the West of Scotland (UWS) United Kingdom - where is involved in researching application of machine learning for protecting computer and IoT networks from modern day cyber attacks. He completed his post graduated in Cisco Networks from the University of Sunderland, United Kingdom. His major research interests comprise software-defined networking (SDN) based network security application with machine learning methods, mainly DDoS attacks traffic classification and detection via packets analysis, packets sampling and entropy calculation.

Zeeshan Pervez is with the University of the West of Scotland (UWS) as an Associate Professor (Reader in the UK). He is a Senior Member of IEEE, Fellow of Higher Education Academy (UK), and a Full Member of EPSRC Peer Review College (UK). His areas of technical expertise are large-scale data analysis, data stream processing, internet-of-things (IoT), cybersecurity, and cloud computing. He has published over 70 indexed journals, peer-reviewed conferences and book chapters. He is actively involved in various UK/EU and international funded projects. He has a track record of securing substantial funding and delivering projects funded through H2020, Erasmus+, Innovate UK, Knowledge Transfer Partnership (KTP), Microsoft Asia, Microsoft Research, Scottish Funding Council, Ministry of Knowledge Economy (South Korea), to name a few.

Keshav Dahal is a Professor in Intelligent Systems and the leader of the Artificial Intelligence, Visual Communication and Network (AVCN) Research Centre at the University of the West of Scotland. Prior to this he was with the University of Bradford and University of Strathclyde UK. He obtained his PhD and MSc degrees from Strathclyde. His research interests lie in the areas of applied AI, trust and security modelling in distributed systems, and scheduling/optimization problems. He has published extensively in his research fields with award winning papers, and has sat on organizing/program committees of over 60 international conferences including general chair or program chair of five IEEE sponsored conferences. He is a senior member of the IEEE.

Ali Kashif Bashir is a Senior Lecturer at School of Computing, Mathematics, and Digital Technology, Manchester Metropolitan University, United Kingdom. He is a senior member of IEEE and Distinguished Speaker of ACM. His past assignments include: Associate Professor of Information and Communication Technologies, Faculty of Science and Technology, University of the Faroe Islands, Denmark; Osaka University, Japan; Nara National College of Technology, Japan; the National Fusion Research Institute, South Korea; Southern Power Company Ltd., South Korea, and the Seoul Metropolitan Government, South Korea. He received his Ph.D. in computer science and engineering from Korea University, South Korea. MS from Ajou University, South Korea and BS from University of Management and Technology, Pakistan. He is author of over 80 peer-reviewed articles. He is supervising/co-supervising several graduate (MS and PhD) students. His research interests include internet of things, wireless networks, distributed systems, network/cyber security, cloud/network function virtualization, etc. He is serving as the Editor-inchief of the IEEE FUTURE DIRECTIONS NEWSLETTER. He is editor of several journals and also has served/serving as guest editor on several special issues in journals of IEEE, Elsevier, and Springer. He has served as chair (program, publicity, and track) chair on several conferences and workshops. He has delivered several invited and keynote talks, and reviewed the technology leading articles for journals like IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE Communication Magazine, the IEEE COMMUNICATION LETTERS, IEEE Internet of Things, and the IEICE Journals, and conferences, such as the IEEE Infocom, the IEEE ICC, the IEEE Globecom, and the IEEE Cloud of Things.

R.Mumtaz (SM'16) received the M.Sc. degree from the Blekinge Institute of Technology, Sweden, and the Ph.D. degree from the University of Aveiro, Portugal. He is currently a Senior Research Engineer with the GS-Lda, where he is involved in EU funded projects. He has authored several conferences, journals, and books publications. His research interests include MIMO techniques, multihop relaying communication, cooperative techniques, cognitive radios, game theory, energy efficient framework for 4G, position information-assisted communication, and joint PHY and MAC layer optimization in LTE standard.

J.Gonzalez (M'04–SM'13) received the master's and Ph.D. degrees in electronic and electrical engineering from the University of Surrey, U.K., in 1998 and 2004, respectively. In 2014, he became a Researcher with the GS-Lda, 1148. He has served as a Project Coordinator for major international research projects (Eureka LOOP, FP7 C2POWER), whilst acting as the Technical Manager for FP7 COGEU and FP7 SALUS. He is currently leading the H2020-ETN SECRET project. In 2009, he was an Assistant Professor with the Universidade de Aveiro, where he was granted as an Associate Professor in 2015. He has been a Chartered Engineer since 2013. He has been a Professor of mobile communications with the University of South Wales since 2017. He was a fellow of IET in 2015.

Raja Majid Ali Ujjan

Zeeshan Pervez

Keshav Dahal

Ali Kashif Bashir

R.Mumtaz

J.Gonzalez